**Mateusz NIEMOJEWSKI, Piotr SAPIECHA**
POLITECHNIKA WARSZAWSKA, INSTYTUT TELEKOMUNIKACJI

# Projekt i implementacja sterowników karty graficznej VGA w układach FPGA

**Inż. Mateusz NIEMOJEWSKI**

Ulończył naukę w XXVII L.O. Im Tadeusz Czackiego w Warszawie. Stopień inżyniera otrzymał na Politechnice Warszawskiej na Wydziale Elektroniki i Technik Informacyjnych w Instytucie Telekomunikacji.

*e-mail: m.niemojewski@gmail.com*

**Dr Piotr SAPIECHA**

Stopień magistra otrzymał na Uniwersytecie Warszawskim na wydziale Matematyki Mechaniki Informatyki na kierunku Matematyka Teoretyczna a następnie pracował na tym wydziale w zakładach: Logiki Matematycznej, Analizy Algorytmów. Równocześnie zaczął pracę na Politechnice Warszawskiej na Wydziale Elektroniki i Technik Informacyjnych w Instytucie Telekomunikacji gdzie uzyskał stopień doktora.

*e-mail: sapiecha@tele.pw.edu.pl*

**Streszczenie**

Celem pracy jest zaprojektowanie interfejsu graficznego i tekstowego umożliwiającego prezentacje informacji na ekranie monitora *VGA* wykorzystując technologie logicznych układów programowalnych, napisanie sterowników sprzętu dla systemu operacyjnego *µClinux* oraz przeprowadzenie i analiza wyników testów uzyskanego rozwiązania ze względu na parametry: prędkość rysowania obrazu, stopień obciążenia pamięci.

**Słowa kluczowe**: VGA, Video Graphics Adapter, FPGA, Field Programable Gates Array, Altera, Stratix II, Lancelot VGA.

## Design and implementation of VGA graphics card in FPGA technology

**Abstract**

The paper presents design and implementation of the text and graphics interface for FPGA based system. The article describes VHDL module and video graphics driver for µClinux operating system. The article describes tests of the device. The paper presents possible future work for the design.

**Keywords**: VGA, Video Graphics Adapter, FPGA, Field Programable Gates Array, Altera, Stratix II, Lancelot VGA.

## 1. Wprowadzenie

Logiczne układy programowalne zyskują coraz więcej zwolenników i są coraz częściej stosowane w różnego typu urządzeniach. Układy te oprócz zastosowań projektowych i specjalizowanych [1, 2, 3, 7] takich jak urządzenia sieciowe lub sterowniki [8], zaczynają pojawiać się w elektronice ogólnego przeznaczenia. Powstają również wersje układów, które nadają się do wykorzystania w urządzeniach przenośnych.

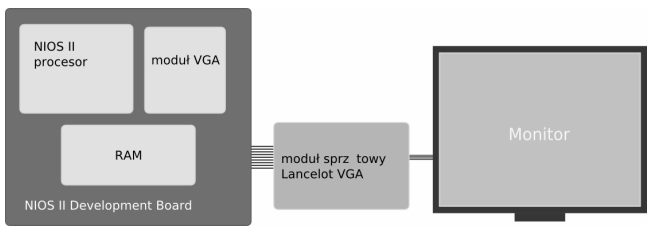
W przypadku zastosowania programowalnych układów logicznych w bardziej powszechnych i uniwersalnych urządzeniach, istotna jest możliwość skorzystania z uniwersalnego interfejsu użytkownika. Stworzenie takiego interfejsu pozwoli na zmniejszenie złożoności układów elektronicznych przez zintegrowanie funkcjonalności sterownika graficznego wewnątrz logicznego układu programowalnego. Poniższa praca stanowi opis implementacji uniwersalnego interfejsu graficznego dla logicznych układów programowalnych z wykorzystaniem platformy *Nios II Development Kit Stratix II Edition* firmy *Altera*.

## 2. Cel

Zadaniem uniwersalnego interfejsu graficznego jest prezentacja pewnych informacji poprzez wykorzystanie w tym celu wizualizacji graficznej. Interfejs taki, może przedstawiać informacje za pomocą tekstu lub obrazu. Uniwersalny interfejs użytkownika powinien posiadać następujące cechy: możliwie nieskomplikowana konstrukcje aby dał się zaimplementować w programowalnych układach logicznych oraz możliwie prosty interfejs użytkownika (*API - application interface*) aby wykorzystanie go nie powodowało niepotrzebnego nakładu pracy. Atutem uniwersalnego interfejsu graficznego może być wykorzystanie istniejących standardów interfejsu użytkownika. Dzięki temu będzie możliwe wykorzystanie wielu dostępnych narzędzi (narzędzi *open source*). Sposobem zapewnienia powyższej funkcjonalności jest przedstawiony projekt.

## 3. Opis projektu

Jako platformę sprzętową wykorzystano w projekcie procesor programowy *Nios II* uruchomiony na układzie z rodziny *Stratix II* firmy *Altera*. Dodatkowo do magistrali procesora podłączony jest moduł napisany w *VHDL* sterujący modułem *Lancelot VGA* będącym cyfrowo-analogowym konwerterem sygnału dopasowującym sygnał do standardu *VGA*. Elementem programowym rozwiązania jest system operacyjny *µClinux* z napisanym sterownikiem urządzenia karty graficznej. Rozwiązanie takie pozwala uzyskać pełną funkcjonalność uniwersalnego interfejsu graficznego. Schemat blokowy platformy sprzętowej przedstawiony jest na rys. 1.

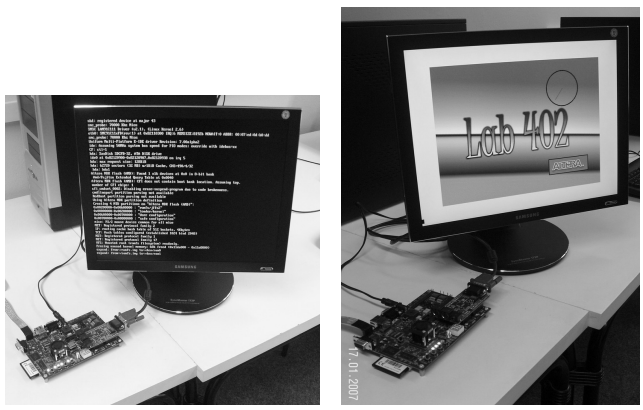



Rys. 1. Schemat blokowy platformy deweloperskiej
Fig. 1. A development platform block diagram

Dzięki wykorzystaniu w projekcie systemu operacyjnego została osiągnięta możliwość łatwego wykorzystana interfejsu graficznego. Umożliwia to wykorzystanie go poprzez napisanie jedynie programu realizującego typowy interfejs programowy. W pracy został wykorzystany system operacyjny *µClinux* bazujący na systemie operacyjnym *Linux*. Dzięki temu użytkownik otrzymuje dostęp do szeregu standardowych aplikacji jak również do wielu narzędzi pomagających przygotować aplikacje.

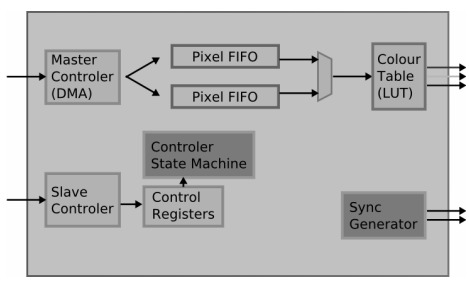
Naturalnym sposobem prezentacji informacji w projekcie jest forma graficzna. Dane przedstawiane na ekranie są reprezentowane przez dwuwymiarowa tabelę w pamięci zawierającą opis każdego punktu rysowanego na ekranie. Funkcjonalność interfejsu tekstowego jest zaimplementowana programowo w sterowniku karty graficznej, dzięki czemu możliwe jest wykorzystanie interfejsu graficznego jak typowej konsoli tekstowej. Na projekt składa się moduł kontrolera grafiki napisany w języku *VHDL* oraz ste-

rownik karty graficznej dla systemu *µClinux* napisany w języku *C*. Moduł kontrolera grafiki jest elementem umożliwiającym wyświetlenie ciągu danych opisujących zawartość ekranu na monitorze. Dane te są przekazywane jako tablica zwierająca numery indeksów tablicy kolorów dla każdego punktu ekranu. Sterownik karty graficznej dla systemu *µClinux* umożliwia wykorzystanie modułu kontrolera grafiki jako urządzenia *framebufer*. Karta graficzna jest dostępna jako standardowe urządzenie systemu. Sterownik ten umożliwia otwarcie urządzenia jako plik oraz mapowanie pamięci kontrolera grafiki w pamięci programu.




Rys. 2. Działająca aplikacja w trybie tekstowym i graficznym
Fig. 2. The working application in text and graphics mode

Moduł kontrolera grafiki jest elementem sterującym modułu sprzętowego *Lancelot VGA*. Przed kontrolerem stoją zadania generowanie sygnałów synchronizacji, odczytywanie danych obrazu z pamięci komputera oraz generowanie sygnału sterującego kanałami kolorów. Schemat blokowy układu kontrolującego działanie karty graficznej przedstawiony został na rys. 3.




Rys. 3. Schemat blokowy układu kontrolujacego działanie karty graficznej
Fig. 3. Block diagram of VHDL module

Zadaniem *sterownika karty graficznej* jest uwidocznienie w systemie operacyjnym i umożliwienie wykorzystania sterownika ekranu przez uruchomione programy. W systemie *Linux* urządzenia umożliwiające wyświetlanie grafiki implementują interfejs zwany *framebuffer*. Sterownik karty graficznej modułu *Lancelot VGA* implementuje interfejs *framebuffer*, ponieważ pozwala to na wykorzystanie szeregu standardowych rozwiązań dostępnych w jadrze systemu, umożliwia również uruchomienie wielu istniejących programów wykorzystujących ten mechanizm.

## 4. Testy

Celem pracy jest przetestowanie możliwości zbudowanego projektu. Istotne jest zbadanie wydajności karty graficznej pod wieloma względami. Zaprojektowane urządzenie zostało poddane szeregowi testów. Część testów badało wydajność interfejsu graficznego a część interfejsu tekstowego. Wszystkie przeprowadzone testy badały szybkość wykonywania operacji.

W celu ujednolicenia programów testujących i wyników pomiarów zastosowany został szkielet programu testującego na którym opierają się wszystkie przeprowadzane testy. Szkielet programu testującego składa się z trzech części: inicjalizującej, testującej i wyświetlającej wyniki. Przed i po części testującej zapamiętywany jest czas systemowy i na jego podstawie dokonywane obliczenia wydajności. Dla każdego z przeprowadzonych testów została przygotowana odpowiednia funkcja testująca. Zadania funkcji testujących przedstawione zostały w tab. 1.

Tab. 1. Zadania funkcji testujących dla przeprowadzonych testów
Tab. 1. Test Functions

| TEST | FUNKCJA TESTUJĄCA |
|---|---|
| TESTY MOŻLIWOŚCI INTERFEJSU GRAFICZNEGO | |
| prędkość zapisywania całego ekranu | funkcja czyszcząca ekran losowym kolorem |
| prędkość rysowania prostokątów na ekranie | funkcja rysująca losowy prostokąt |
| prędkość rysowania linii na ekranie | funkcja rysująca losowa linie |
| prędkość rysowania punktów na ekranie | funkcja rysująca losowy punkt |
| prędkość odczytu danych z obszaru pamięci ekranu | funkcja odczytująca losowy punkt z pamięci obrazu |
| prędkość odczytu danych z poza obszaru pamięci ekranu | funkcja odczytująca losowy punkt z poza pamięci obrazu |
| TESTY MOŻLIWOŚCI INTERFEJSU TEKSTOWEGO | |
| prędkość rysowania znaków | funkcja zapisująca znak w strumieniu konsoli |
| prędkość przesuwania tekstu w górę ekranu | funkcja zapisująca linie tekstu w strumieniu konsoli |
| prędkość czyszczenia ekranu i zapisania znakami | funkcja zapisująca kod czyszczenia ekranu terminala a następnie zapisująca zadana ilość znaków w strumieniu konsoli |

Pierwsze testy są typowymi testami kart graficznych i mogą posłużyć do porównania osiągnięć układu z innymi kartami graficznymi. Testy odczytu danych z różnych obszarów pamięci maja na celu określenie obciążenia magistrali pamięci przez układ wyświetlania grafiki. Testy te są dla porównania przeprowadzone na tym samym układzie ale z wyłączoną karta graficzna oraz na układzie bez karty graficznej. Testy interfejsu tekstowego są przeprowadzone w celu analizy wydajności. Testy te maja na celu określenie jaki jest najefektywniejszy sposób korzystania z tego interfejsu.

Z pomiarów wynika, że szybkość rysowania nie zależy od ilości przeprowadzonych prób. Uzyskane wyniki różnią się bardziej znacząco od średniej przy najmniejszej ilości powtórzeń testów. W tab. 2 zamieszczone zostały uśrednione wyniki pomiarów. W celu uśrednienia wyników pomiarów interfejsu graficznego został odrzucony pomiar dla najmniejszej ilości testów.

Wynik prędkości zapisania całego ekranu o wartości 0,78 obrazu na sekundę pokazuje, że przedstawione rozwiązanie nie nadaje się do prezentowania płynnych animacji ponieważ aby animacja była odbierana płynnie potrzebna jest szybkość co najmniej 16 obrazów na sekundę. Pod względem animacji większe nadzieje daje wynik rysowani prostokątów na ekranie. Daje on możliwość, że przy pewnej wielkości obrazu można osiągnąć wartość około 10 prostokątów na sekundę, co daje w efekcie poklatkową animacje. Pod względem szybkości rysowania linii można osiągnąć pewna płynność animacji wektorowej, przy założeniu, że obraz

będzie się składał z kilku linii. Możliwe jest stworzenie aplikacji rysującej działający zegar.

Tab. 2. Uśrednione wyniki badań
Tab. 2. Average Measured Results

| TEST: PRĘDKOŚĆ ZAPISYWANIA CAŁEGO EKRANU | |
|---|---|
| Wynik [1/s] | 0,78 |
| TEST: PRĘDKOŚĆ RYSOWANIA PROSTOKĄTÓW NA EKRANIE | |
| Wynik [1/s] | 6,35 |
| TEST: PRĘDKOŚĆ RYSOWANIA LINII NA EKRANIE | |
| Wynik [1/s] | 18,00 |
| TEST: PRĘDKOŚĆ RYSOWANIA PUNKTÓW NA EKRANIE | |
| Wynik [1/s] | 6.854,61 |
| TEST: PRĘDKOŚĆ ODCZYTU DANYCH Z OBSZARU PAMIĘCI EKRANU | |
| Wynik [1/s] | 9.965,63 |
| TEST: PRĘDKOŚĆ ODCZYTU DANYCH Z POZA OBSZARU PAMIĘCI EKRANU | |
| Wynik [1/s] | 10.008,26 |
| TEST: PRĘDKOŚĆ ZAPISYWANIA ZNAKÓW | |
| Wynik [1/s] | 1.962,87 |
| TEST: PRĘDKOŚĆ PRZESUWANIA TEKSTU W GÓRĘ EKRANU | |
| Wynik [1/s] | 3,07 |
| TEST: PRĘDKOŚĆ ZAPISYWANIA EKRANU TEKSTU | |
| Wynik [1/s] | 1,40 |

Porównując prędkość odczytu z różnych obszarów pamięci można zauważyć, że prędkość odczytu z obszaru pamięci karty graficznej jest w przybliżeniu taka sama jak prędkość odczytu z obszaru poza pamięcią karty graficznej. Jest to spodziewany wynik ponieważ obszary te znajdują się na tej samej magistrali pamięci i oba odczyty są ograniczane przez dostęp karty graficznej do pamięci obrazu. Niespodziewanym jest za to gorszy dostęp do pamięci która podłączona jest do oddzielnej magistrali pamięci. Przyczyną tego spowolnienia jest konstrukcja płyty deweloperskiej na której zbudowany jest projekt. Moduł pamięci *SRAM* który nie jest mapowany w przestrzeni pamięci operacyjnej znajduje się na tej samej magistrali co pamięć *Compact-Flash* i urządzenie karty sieciowej. Z tego powodu dostęp do pamięci *SRAM* jest wolniejszy niż do pamięci operacyjnej.

Porównując wyniki zapisania ekranu tekstowego i przesuwania linii obrazu widać wyraźnie, że projektując aplikacje tekstowa, należy wziąć pod uwagę, że zapisywanie obrazu jest znacznie szybsze niż przesuniecie ekranu o trzy linie. Należy więc w programach których komunikaty tekstowe mogą być dłuższe niż trzy linie tekstu zastosować mechanizm wyświetlania komunikatów przez przerysowanie ekranu tekstu. Tak duża różnica w prędkości wynika z faktu, że terminal tekstowy jest obsługiwany programowo i w celu przesunięcia obrazu o każda linię niezbędne jest przepisanie zawartości całego ekranu. Szybkość przesuwania ekranu tekstowego jest jednak większa niż szybkość przerysowania obrazu jako grafiki, ponieważ w trybie konsoli tekstowej wbudowano mechanizm nie przerysowywania znaków jeśli w linii powyżej znaj jest taki sam.

## 5. Podsumowanie

W pracy przedstawiona została implementacja uniwersalnego interfejs graficznego z wykorzystaniem platformy *Nios II Development Kit Stratix II Edition* firmy *Altera*. Zaprezentowane rozwiązanie pozawala na stworzenie zarówno graficznego jak i tekstowego interfejsu użytkownika. W projekcie został omówiony moduł karty graficznej. Zbudowane urządzenie przetestowano. Zostały przeprowadzone serie testów badających wydajność rozwiązania. Omówione zostały wyniki z których można wnioskować, że interfejs może być wykorzystany w przypadku gdy na obrazie nie dokonuje się wielu szybkich zmian lub animacji. Projekt pozwala na wprowadzenie usprawnień jednak w tym celu należy skonstruować układ przygotowany specjalnie dla potrzeb projektowania interfejsu graficznego.

Przyszła praca nad projektem ma na celu poprawienie wydajności urządzenia. W celu zwiększenia wydajności interfejsu tekstowego należy zaprojektować rozwiązanie sprzętowe wspomagające rysowanie znaków tekstowych. Zwiększenie wydajności interfejsu graficznego może zostać osiągnięte przez przygotowanie nowego układu projektowego w którym zostanie wykorzystana dwu portowa pamięć dla karty graficznej.

Praca naukowa finansowana ze środków na naukę w latach 2007-2010 jako projekt badawczy nr N517 003 32/0583.

This work was supported by the Ministry of Science and Higher Education of Poland - research grant no. N517 003 32/0583 for 2007-2010.

## 6. Literatura

[1] P. Lysaght, W. Rosenstiel, „New Algorithms, Architectures and Applications for Reconfigurable Computing”, Springer, 2005.
[2] N. S. Voros, K. Masselos, „System Level Design of Reconfigurable Systems-on-Chip”, Springer, 2005.
[3] M. B. Gokhale, P. S. Graham, „Reconfigurable Computing Accelerating computation with field-programable gate arrays”, Springer, 2005.
[4] A. Rubini, J. Corbet, „Linux Device Drivers, 2nd Edition”, O'Reilly, 2001.
[5] Altera Corporation , http://www.altera.com
[6] Microtronix Datacom Ltd. , http://www.microtronix.com/
[7] FPGA w superkomputerach :
- http://www.cray.com/
- http://www.starbridgesystems.com/hypercomputing/

[8] FPGA w Telekominikacji :
- Cisco 2600 Series Router, http://www.cisco.com/
- Voyager Star Video Portable Conference System, http://www.aethra.com/
- EXM1 Escape Flexible Modem, http://www.escapecom.com/
- MXP Platform, http://www.motorola.com/
- Canopy, http://www.motorola.com/

[9] FPGA w zastosowaniach militarnych :
- http://www.meecc.com/presentations/CIUFO.pdf

[10] FPGA i VGA :
- http://www.fpga4fun.com/PongGame.html,
- http://www.bridgeport.edu/sed/projects/cpe447/pingpong/pingpong.html,
- http://www.fpga.ch/samples/vga.php, http://www.opencores.org/projects.cgi/web/vga_lcd/overview

*Artykuł recenzowany*